8. Marta

Program Name: Marta.java **Input File: marta.dat**

Marta has been studying hard for the upcoming UIL season but is having some difficulty with the classic bubble sort for arrays. She understands that every pass through the array will move smaller items one direction and larger items the other direction. An ascending sort will move smaller items towards the lower or [0] array position while larger items move towards the higher-end of the array. A descending sort would move items the opposite directions.

She would like to be able to confirm her manual sort progress after a specific number of passes to view how the arrangement should be changing. Here is an example:

Initial	95	11	68	13	33	26	24	60	56	47	79	21
Pass 1	11	68	13	33	26	24	60	56	47	79	21	95
Pass 2	11	13	33	26	24	60	56	47	68	21	79	95
Pass 11	11	13	21	24	26	33	47	56	60	68	79	95

However, the sort may not get to the requested pass count, depending on how quickly the numbers settle into their correct positions. In that case, the numbers will be fully sorted but Marta wants to know how many passes it actually took to complete the sort when that happens.

Can you help Marta modify her sort to output a list of numbers a specific pass of the sort?

Input: First line will contain an integer T with $1 \leq T \leq 10$, the number of test cases. Each test case will consist of two lines of space-separated integer data. First line will contain an integer N with $10 \leq N \leq 50$, the number of integers to be sorted, followed by another integer P with $1 \leq P < N$, the sort pass after which the array content will be displayed. The next line will contain N integers in range [10, 99], the data to be sorted into ascending order but displayed after only P or fewer passes have been completed.

Output: Each test case will produce 1 line of output that starts with "`Test case #1 pass #2: `" where `#1` is the test case number and `#2` is the actual number of passes completed which can be less than P because of the early exit technique from a proper bubble sort.

Sample input:

```
3
12 1
95 11 68 13 33 26 24 60 56 47 79 21
12 2
95 11 68 13 33 26 24 60 56 47 79 21
25 23
71 40 52 66 95 80 61 56 26 64 34 47 99 51 96 86 63 55 49 72 82 43 93 42 76
```

Sample output: (*lines that are indented are continuation of previous line*)

```
Test case 1 pass 1: 11 68 13 33 26 24 60 56 47 79 21 95
Test case 2 pass 2: 11 13 33 26 24 60 56 47 68 21 79 95
Test case 3 pass 21: 26 34 40 42 43 47 49 51 52 55 56 61 63 64 66 71 72 76 80 82 86 93
   95 96 99
```